Scheduling Jobs in Flowshops with the Introduction of Additional Machines in the Future

Dongchen Lu and Rasaratnam Logendran*

School of Mechanical, Industrial, and Manufacturing Engineering

Oregon State University, Corvallis, Oregon 97331-6001, USA

Lud@onid.orst.edu, Logen.Logendran@oregonstate.edu

*Corresponding author: Rasaratnam Logendran, Tel: 1-541-737-5239, Fax: 1-541-737-2600
E-mail:Logen.Logendran@oregonstate.edu

Scheduling Jobs in Flowshops with the Introduction of Additional Machines in the Future

The problem of scheduling jobs to minimize total weighted tardiness in flowshops, with the possibility of evolving into hybrid flowshops in the future, is investigated in this paper. As this research is guided by a real problem in industry, the flowshop considered has considerable flexibility, which stimulated the development of an innovative methodology for this research. Each stage of the flowshop currently has one or several identical machines. However, the manufacturing company is planning to introduce additional machines with different capabilities in different stages in the near future. Thus, the algorithm proposed and developed for the problem is not only capable of solving the current flow line configuration but also the potential new configurations that may result in the future. A meta-heuristic search algorithm based on Tabu search is developed to solve this NP-hard, industry-guided problem. Six different initial solution finding mechanisms are proposed. A carefully planned nested split-plot design is performed to test the significance of different factors and their impact on the performance of the different algorithms. To the best of our knowledge, this research is the first of its kind that attempts to solve an industry-guided problem with the concern for future developments.

Keywords: total weighted tardiness; tabu search; nested split-plot design, additional machines

1. Introduction

This research is guided by a real industrial scenario. The supplier takes orders from customers, who have a priority demand to receive the products on time. This drives the supplier to minimize the tardiness. Moreover, the supplier is taking orders from different customers, some of whom give a higher penalty when their orders are delayed, while the others give a lower penalty under the same circumstances. Thus, a weight is introduced in this problem to represent different importance for the supplier to finish processing the jobs on time.

The research on scheduling problems has grown considerably in the past decades. Most research assumes that the sequence of jobs has no effect on the setup time required on a machine. However, in some cases the setup time on a machine changes significantly when transferring from one job to another. For the supplier in this research, the setup time is sequence-dependent, meaning that the setup time on every machine depends on both the current and the immediately preceding jobs.

Because the supplier gathers raw materials from a warehouse which releases the materials at different times, dynamic job release times are assumed in this research, meaning that the jobs can be released at any given time during the current planning horizon (the number of jobs is predetermined). At the same time, the supplier's flow line can be quite busy. It means that at the start of a current planning horizon, a subset of machines has a chance of processing jobs released in the previous planning horizon. Thus, dynamic machine availability is also assumed in this research.

All of the jobs may not need to be processed through each and every machine in the flowshop. Thus, jobs have a chance to skip one or more stages in the flowshop. Consequently, a machine skipping ratio is introduced in the research. When an order is taken from a customer, the jobs in that order may not require processing on a subset of stages.

Two production lines, namely Bevel line and Spur line, were provided as the machine/stage layouts by the supplier. Each production line has a front end and a back end. The jobs that complete their operations on the front end machines spend an estimated amount of time outside, prior to arriving for their operations on the back end machines. Thus, there would be four flowshops that need to be investigated in this research. However, Spur front end and Bevel back end have the same machine configuration, which narrows the research down to three flowshops. To further generalize the problem structure, a combined flowshop is used to demonstrate the problem. Figure 1 shows the layout of this hybrid line. There are a total of six stages in the generalized flow shop. Except for stage 3, which has two machines currently in that stage, all other stages have one machine.

Machine and stage switches are introduced in this layout in order to represent different flowshops in the actual problem. As mentioned before, there are three types of flowshops, here we name them as flowshops A, B, and C. Flowshop A consists of three stages, so stages 4, 5, and 6 are switched off in the generalized layout, which represents Spur front end and Bevel back end. Also, flowshop A has only one machine in stage 3, so M_{32} is switched off for flowshop A. Flowshop B includes six stages, representing Spur back end, which only has a single machine in stage 3, so M_{32} is switched off for flowshop B. Flowshop C has five stages, representing Bevel front end, which has two identical machines in stage 3, so only stage 6 is switched off for flowshop C.

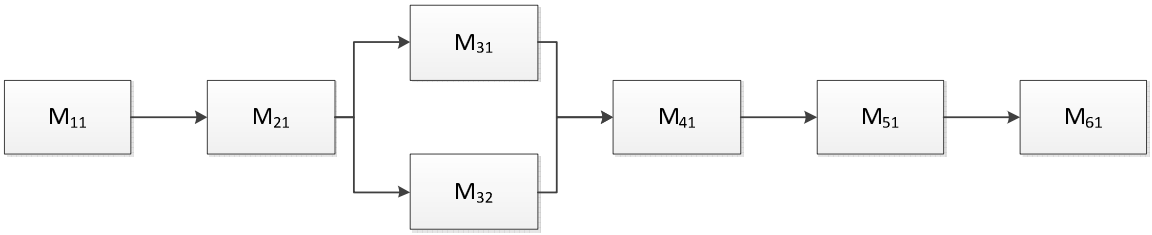

Figure 1 Hybrid-line layout

The most challenging part of this research is that the supplier company is growing because of their growing customer base so that it may introduce new machines in different stages in order to increase the productivity of their manufacturing operations in the near future. In terms of developing a scheduling algorithm, the company requests to have a long term product that can not only meet the company's scheduling tasks now but also in the future when new machines are introduced. Based on the company's estimation, each stage stands a chance of being introduced with at most one additional machine in the future. Thus, for each flowshop, there are several potential machine configurations. The machines introduced may not be identical to the original machines in each stage, which makes the future flowshop a hybrid flowshop. The complexity of the problem increases significantly when this feature is incorporated. For example, flowshop A is currently a 3-stage flowshop with one machine in each stage. However, in the future, flowshop A can be a hybrid flowshop with parallel machines in stages 1 and 3. To the best of our knowledge, no previous research of this kind has been reported in the published literature.

Following the three field notation of the scheduling problems by Lawler et al. (1993), Chen et al. (1998), and Allahverdi et al. (1999), we designate this problem as $FHm|r_j, \bar{d}_j, skip, ST_{sd}|\sum(w_j)T_j$. The rest of this paper is organized as follows. In Section 2, a review of previous work on flexible flowshop tardiness minimization problems is performed. The tabu search algorithm developed for this research is presented in Section 3. The data generation mechanism for example problems is demonstrated in Section 4. In Section 5, the application of the search algorithm to an example problem is shown. Section 6 reports the performance of the search algorithm based on statistical test results from experimental design. Finally, in Section 7, we conclude the result of this research and point out the direction for future research.

2. Literature review

The earliest available heuristic algorithm for solving the parallel machine scheduling problem is to order the jobs according to the earliest due date (EDD) rule and assign them to the machines afterwards (Wilkerson and Irwin, 1971). Dogramaci and Surkis (1979) apply three different priority rules, namely EDD, shortest processing time (SPT) and minimum slack. However, since the single machine problem is NP-hard, this procedure is only good for relatively small instances. Sequence-dependent setup time reflects the similarity and dis-similarity among jobs, which fits with the real problems in industries. The current machine-stage layouts of two flowshops in this research are having one machine in each stage. In this kind of flowshop, Rubin and Ragatz (1995) developed a genetic search algorithm to minimize total tardiness for a job scheduling problem with sequence-dependent setup time. A Simulated Annealing algorithm was developed by Tan and Narasimhan (1997) to minimize total tardiness for a single-stage problem with parallel machines. Lee et al. (1997) proposed a three-phase heuristic for the problem of minimizing the total weighted tardiness on a single machine in the presence of sequence-dependent setup times. Tabu search is widely used in hybrid flowshop scheduling problems based on the review by Allahverdi et al.(1999), who also pointed out the importance of mimicking real industry scenarios. Sen et al. (2003) reviewed various available solution approaches to solve the parallel machine job scheduling problem to minimize tardiness. They concluded that the focus should be placed more on heuristics and the development of search

algorithms. Although the flexible flowshop problem has been widely studied in the literature, most of the studies related to flexible flow shop problems are concentrated on problems with identical machines when minimizing tardiness (Gupta et al. (2002) and Alisantoso et al. (2003)). A generalized hybrid flowshop scheduling problem was investigated by Ruiz and Maroto (2006). In spite of introducing machine eligibility, they assume static job releases and machine availabilities, which may not be applicable in industry practice. Shim and Kim (2007) developed a branch-and-bound algorithm to minimize total tardiness in a flowshop with identical-parallel machines with static job release and machine availability. In the real world, it is more common that a newer machine would be placed next to an older machine, which is why we assume that unrelated-parallel machines will be introduced in the future. Dynamic job release and machine availability are another industry-relevant requirements that often found missing in previous research. In a job scheduling problem with sequence-dependent setup times and unrelated-parallel machines in a single stage, Bilge et al. (2004) assumed dynamic job releases (but not dynamic machine availability) when minimizing total tardiness. Logendran et al (2007) proposed a tabu-search-based algorithm in their study of minimizing the weighted tardiness of jobs on unrelated-parallel machine scheduling with sequence-dependent setups. In their study, the dynamic release of jobs and dynamic availability of machines were assumed. Biskup et al. (2008) proposed a new heuristic approach to the problem of scheduling a given number of jobs on a specified number of identical parallel machines to minimize total tardiness. Zhang et al. (2011) developed a genetic algorithm for solving the flexible job-shop scheduling problem to minimize makespan based on static assumptions. Most of the previous works mentioned above investigate static machine-stage layouts, but this research focuses on providing implementable solutions in industry practice motivated by hybrid flowshops that can have multiple unrelated-parallel machines in one or more stages in the future with dynamic assumptions.

3. Search algorithm

The proposed problem consists of considerable flexibility and complexity. The simplest scenario of the proposed problem would be a multi-machine flow shop scheduling problem with dynamic job release and machine availability times and sequence-dependent

setup time, which is proven to be NP-hard. As machine skipping and unrelated parallel machines are introduced, the complexity of the problem is further increased. A search algorithm based on tabu search (TS) is developed to solve the problem and produce near optimal solution. The algorithm developed in this research, to fulfill the requirement of the company, is capable to solve problems with the machine configurations now and the ones will occur in the future based on the estimations from the company. The idea of using tabu seach for solving hard combinatorial optimization problems was first introduced by Glover (1986). Heuristics based on tabu search have been proven to be successful in previous work on scheduling problems (Logendran and Sonthinen (1997); Logendran and Subur (2004); Logendran et. al (2007)). With the adaptive memory structures, tabu search is able to provide satisfying results in combinatorial optimization problems.

An initial solution (IS) is needed to trigger the search. For job scheduling problems, an IS should be a given sequence of jobs. We propose 6 ISs to trigger the TS, as described below:

IS1: Earliest due date (EDD). EDD is a widely used sequencing rule in scheduling problems. Jobs are sorted by their due date in an increasing order.

IS2: Highest weighted tardiness on the last stage (HWT-Max). The estimated tardiness on the last stage is used in the HWT-Max rule. The formula below is used to calculate tardiness as:

$$t_{max,j} = Max\{0, C_{max,j} - d_j\} \quad (1)$$

When a job *j* has an estimated completion time ($C_{max,j}$) that is larger than its due date (d_j), it also has a non-zero estimated tardiness ($t_{max,j}$). The jobs are sorted by their estimated weighted tardiness ($w_j \cdot t_{max,j}$, w_j denotes the weight of job *j*) from high to low, because a job with a larger estimated weighted tardiness tends to be processed earlier.

IS3: Due date to weight ratio. IS3 is based on IS1. Weights of jobs are brought into the picture to indicate that a job with a higher ratio should receive a bigger penalty when

delayed. The jobs are sorted by the ratio d_j/w_j from low to high, because a job with a smaller due date and a bigger weight tends to be processed earlier.

IS4: Hybrid critical ratio. Compared to IS1 and IS3, IS4 introduces more information to gain insights about the jobs. The jobs are sorted by the ratio $d_j/(C_{max,j} \cdot w_j)$ from low to high, because a job with a smaller due date and a larger weight and estimated completion time tends to be processed earlier.

IS5: Due date to estimated time in system ratio. IS5 tries to look into the problem in a different angle by considering a job's estimated time in system instead of $C_{max,j}$. A job's estimated time in system (TIS_j) is computed by the formula below:

$$TIS_j = \sum_{i=1}^{n} r_{ij} + \sum_{i=1}^{n} \beta \bar{st}_{ij} \quad (2)$$

r_{ij} stands for the run time of job j on stage i. $\bar{st}_{ij}$ stands for the average setup time of job j on stage i. β is introduced as an adjustment for the setup time. The reason of introducing β and the evaluation of it will be demonstrated in Section 4.

IS6: Highest weighted tardiness. IS6 is a more detailed IS finding mechanism compared to IS2. The estimated tardiness of all jobs on all stages is used in this mechanism. Similarly, estimated tardiness of job j in stage i is evaluated using the formula $t_{i,j} = Max\{0, C_{i,j} - d_j\}$. Starting with the 1st stage, the estimated tardiness of each job is evaluated, then jobs with non-zero estimated tardiness are sorted by their estimated weighted tardiness from high to low. In the next stage, jobs that are already sequenced are ignored and only those left, which have non-zero estimated tardiness are sorted and added to the bottom of the existing job sequence. The motivation for IS6 is that a job showing tardiness at the earlier stage tends to be processed earlier.

After IS is identified, the search algorithm takes the IS and starts evaluating and perturbing the IS to move to promising new solutions. Two different neighborhood functions are implemented in this research. Exchange move is performed by selecting two jobs and exchanging their positions. Insert move is performed by selecting one job and one position in the current sequence and inserting that job into the position. The target position

of insert move is selected by referring to a position next to a particular job. Based on a given sequence, all possible moves will be tried. Candidate list (CL) and index list (IL) are recording the potential local optimum and local optimum separately. A sequence is a potential local optimum if it gives a better objective function value than its parent. A sequence is a local optimum if it has no worse objective function value than its parent and child. The sequences stored in CL are also representing the nodes of the search steps. Thus, when a certain move gives a resulting sequence that is already recorded in CL, this move can be ignored and the next move can be immediately performed because going back to the same node of the search will no doubt provide the same result as before and may also lead the search into a circle. Evaluation of the objective function is done after each available move is made. The solution with the best objective function value is selected and the move that resulted in this sequence is recorded in the tabu list (TL). Aspiration level (AL) and TL work together to assist the TS memory function to go after the better solution and move among the solution space. AL records the best objective function value found so far. Moves in the TL will not be accepted in the future perturbation unless the move provides a better objective function value than the best solution found so far. TL is not an endless list. Therefore, after a number of iterations, new moves will replace the old moves that are tabu. The number of moves that can be recorded in TL is known as the tabu list size (TLS). TLS can affect the search a lot because it controls the number of available moves in each iteration. A big TLS can restrict the search because the moves that can be made are limited but it can also force the search to go far away from the starting solution. While a small TLS can relax the search, it can limit the area in the solution space that the search will reach. An appropriate TLS should be decided by the characteristic of the problem. After a number of iterations the search should be terminated. The stopping criterion of the search involves two parameters: Number of iterations without improvement (IWI) and number of entries into the index list (EIL). The search is terminated when no improvement has been made for a certain number of iterations or a certain number of local optima has been found. After the stopping criterion is met, a short-term memory search is finished. Long-term memory can be used to intensify or diversify the search. Long-term memory (LTM) is implemented by introducing a matrix that records the frequency of a position occupied by different jobs when moves are selected during the short term search. In other words, during each short

term memory search, the matrix records the job-position combination after each move is made. After the short term memory search is terminated, this matrix is used to restart the search. To intensify the search, LTM-max is used, which selects the most occupied position with the job that occupied it most and fix them together in the next short term memory search. To diversify the search, LTM-min is used, which selects the least occupied position with the job that occupied it least and fix them together in the next short term memory search. When selecting positions and jobs, a position-wise first best strategy is used to break ties.

4. Data Generation

To test the performance of the initial solution and search algorithm, meaningful test problems need to be generated. The supplier is very particular about maintaining confidentiality of the original data. Thus, the mechanism used in data generation is, to some extent, guided by previous research (Logendran et al. (2007), Pandya and Logendran (2010)). As discussed before each job will have several categories of data: weight, run time, setup time, release time (when the job is ready to start processing on the 1st stage) and due date. The weight of a job is generated uniformly in [1, 3]. A uniform distribution of [1, 40] is used to generate the run time of jobs in all the stages. The same uniform distribution is also used to generate sequence-dependent setup times of jobs. The release time of jobs are generated from an exponential distribution with λ=0.05 (20 minutes per arrival). To mimic the industrial practice more precisely, instead of using the availability time ($a_{i,k}$) generated from exponential distribution directly, accumulated machine availability time ($a'_{i,k}$) is proposed. As the machine availability time increases while going through the stages, we need to update the availability times in each stage. Realistically, the availability time of a machine at each stage should be equal to the completion time of processing the reference job (the last job from the previous planning horizon) at that stage and the ones which were assigned and processed by the machines in the previous stages. Because the computation of the processing times (run time plus setup time) of reference jobs fall outside of the jobs released in the current planning horizon, we propose a mechanism to approximately account for the revision needed for evaluating the machine availability times as follows:

Case 1: one- to- one transfer

In this case, the availability time at stage *i* is the availability time at stage *i* plus the setup time of this machine for processing reference job R_i when preceded by job R_{i-1}. Since we do not know the setup time of reference jobs, we approximately evaluate the value by using the jobs released in the current planning horizon. Thus, we use $\bar{S}_{i,k}$, the average of setup times for all of the jobs in the current planning horizon, to estimate the setup time required for the reference jobs. The updated availability time is calculated as follows by equation (3):

$$a'_{i,k} = max\{\, a'_{i-1,k}\,, a_{i,k} + \bar{S}_{i,k}\} + \bar{P}_{i,k} \quad \text{, where } k=1 \tag{3}$$

$\bar{P}_{i,k}$ is the average run time of all jobs in the current planning horizon on machine *k* at stage *i*.

Case 2: many-to-one transfer

In this case, machine *k* at stage *i* is available only when all the reference jobs at the previous stage are released. In the other words, it is available only after all previous machines are available. The updated availability time is calculated by equation (4):

$$a'_{i,k} = max\{\, \sum_{k=1}^{m_i} a'_{i-1,k}\,, a_{i,k} + \bar{S}_{i,k}\} + \bar{P}_{i,k} \tag{4}$$

Case 3: many-to-many transfer

In contrast with Case 2, in this case the machines at stage *i* do not need to wait for all of the machines at stage *i*-1 to release their jobs because the reference jobs from previous stage *i*-1 can start their processes on parallel machines at stage *i*. Thus, we assume that each machine is available after the maximum availability time of the previous stage. A ratio is used to balance the maximum, as the number of machines (*k*) at stage *i* increases the availability time of those machines decreases because there are more parallel machines that can process the reference jobs. Clearly, decreasing the number of machines (*k'*) at stage *i*-1 will result in increasing the estimated availability time of the machines at stage *i*.

$$a'_{i,k} = max\{\frac{k}{k'} \cdot \max\{a'_{i-1,k}\}\,, a_{i,k} + \bar{S}_{i,k}\} + \bar{P}_{i,k} \tag{5}$$

The due date of a job is generated by taking advantage of the mechanism used by Pandya and Logendran (2010). The use of meaningful due dates can help evaluating the performance of initial solutions and search algorithms. The generation of meaningful due dates is more challenging than other data. τ and R is used as tardiness factor and range factor, respectively. The maximum completion time of all jobs (C_{max}) is used to estimate the meaningful completion time of jobs. τ reflects the tightness of the average due date, given by the equation $\tau = 1 - \bar{d}/C_{max}$, where $\bar{d}$ is the average due date. A large value of τ results in a small $\bar{d}$, which means the due dates are tight and a small value of τ results in a big $\bar{d}$, which means the due dates are loose. R reflects the range of the due dates, given by the equation $R = (d_{max} - d_{min})/C_{max}$ where d_{max} is the largest due date and d_{min} is the smallest due date. A large value of R results in a big difference between d_{max} and d_{min} (a wide range) and a small value of R results in a small difference between d_{max} and d_{min} (a narrow range). Therefore, a combination of τ and R gives a set of due dates with a particular characteristic. The due dates are then generated based on random numbers between 0 and 1. If the random number is from (0, τ], then the due date will be generated in the interval $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$. If the random number is from (τ, 1), then the due date will be generated in the interval $[\bar{d} - R\bar{d}, \bar{d}]$. To estimate the C_{max} on the last stage, equation (6) is proposed:

$$C_{max,i} = \frac{\sum_{j=1}^{n}\left[\frac{\sum_{\substack{k=1 \\ r_{j,i,k}>0}}^{m_i} max\left(e_{j,i}, a_{i,k} + \beta \cdot \bar{s}_{j,i,k}\right) + r_{j,i,k}}{m_i}\right]}{m_i}, e_{j,0} = rt_j \qquad (6)$$

m_i is the total number of machines in stage i. $a_{i,k}$ is the availability time of machine k in stage i. $\bar{s}_{j,i,k}$ denotes the average setup time of job j on machine k in stage i. The run time of job j on machine k in stage i is denoted by $r_{j,i,k}$.The total number of jobs is denoted by n. rt_j denotes the original release time when the job arrives in the flow shop. In reality, a quality schedule would try to changeover from one group to another that requires the smallest setup time in a particular stage. In other words, the use of the average setup time is providing an inaccurate estimate of the completion time for each stage. Depending upon each job's set of sequence-dependent setup times on a particular machine, β is introduced

as an adjustment to the average setup time. To identify a rational value of β, the coefficient of variation (CV) is defined for the sequence-dependent setup times for a group on a machine. $CV = s/\bar{x}$ where s is the sample standard deviation and $\bar{x}$ is the mean. A linear relationship between β and CV is assumed: $CV = 0.01$ corresponds to $\beta = 0.9$ and $CV = 1.0$ corresponds to $\beta = 0.1$. The release time of every job in stage 1 is the original release time of the job (rt_j).

The estimated completion time of a job evaluated for stage i ($C_{j,i}$) serves as the release time of the job in the following stage i+1 ($e_{j,i+1}$). We propose equation (7) to estimate the release time of jobs in the (i+1)th stage:

$$e_{j,i+1} = \frac{\sum_{\substack{k=1 \\ r_{j,i,k}>0}}^{m_i} \max(e_{j,i}, a_{i,k}+\beta\cdot\bar{s}_{j,i,k})+r_{j,i,k}}{mi} \quad (7)$$

A job's estimated completion time is calculated by comparing the job's release time in stage 1 with the machine's availability time combined with the adjusted average setup time. The larger value is selected because the job can start processing only when the job is released from the previous stage and the machine in the current stage is ready to process the job. When a job skips a particular stage, the job's last release time calculated in the stage where the job has an operation will serve as the release time of the next non-skipping stage where the job requires another operation. The estimated completion time of one job on the last stage ($C_{max,j}$) used in IS2 and IS4 is equal to $e_{j,i+1}$ when i equals to the total number of stages. As the job is processed through the stages, C_{max} increases progressively. Constraint (8) shows this fact:

$$C_{max,i+1} \geq C_{max,i} \quad (8)$$

$C_{max,i+1} = C_{max,i}$ will only happen when all jobs skip stage i+1; otherwise $C_{max,i+1} > C_{max,i}$ will hold true. The C_{max} evaluated on the last stage is used to estimate the maximum completion time of all jobs for a given problem.

5. Application of the search algorithm

In order to make sure that the search algorithm can give desirable results to problems with different characteristics, several search parameters (SPs) need to be tuned before running the search. As mentioned above in Section 4, TLS should be tuned because it has an important effect on the search steps. Likewise, IWI and EIL should be tested to determine proper stopping criteria.

To tune the SPs, 7 different problems representing different structures and different sizes are generated. Within each structure, number of jobs, number of machines in stages, and machine skipping ratio are used as characteristic parameters (CPs) which guide the differentiation of problems. And then IWI and EIL are fixed to a large number (25 in this case). The algorithm is tested on each problem with TLS increasing from 1 to 25. The TLS which gives the best result (the solution with the lowest objective function) is recorded. A linear regression is then performed on the best TLSs and the CPs from the set of problems to obtain a formula to evaluate the proper SP in the future. After testing TLS, IWI and EIL are tested similarly to obtain the tuned parameter values.

To generate a machine configuration, a random flow shop type from A, B or C is chosen first. Once a flow shop type is determined, a random number is generated between 0 and 1 for each stage. If the number falls into (0,0.5) the stage will not obtain an additional machine, otherwise the stage will obtain an additional machine. Software DataFit (2008) is used to perform linear regression between SPs and CPs. According to the significance reported by the software, number of jobs (NOJ), number of stages (NOS), number of stages with increased machines (NSIM), and machine skipping ratio (MSR) are used to be the independent variables. The size of the problem is determined by number of jobs. Small problem contains 5-12 jobs. Medium problem contains 13-25 jobs. Large problem contains 26-33 jobs. MSR can vary among 0.1, 0.2, and 0.3. Classified by problem sizes, the parameter evaluating formulae are shown below (every value is rounded normally):

Small problem

TLS= NOJ*0.36+NS*1.3-NSIM*0.48+MSR*6-3
IWI= NOJ*0.52+NS*0.8-NSIM*0.34+MSR*3.5
EIL= NOJ*0.5+NS*1.2+NSIM*0.7-MSR*4.2-2
TLS+= NOJ*0.36+NS*1.5-NSIM*0.41+MSR*6-4
TLS-= NOJ*0.36+NS*1.1-NSIM*0.4+MSR*5.1-5

Medium problem

TLS= NOJ*0.31+NS*1.3-NSIM*0.5+MSR*5-6.6
IWI= NOJ*0.76+NS*0.8-NSIM*1.1+MSR*2.9
EIL= NOJ*0.65+NS*0.77+NSIM*0.3-MSR*3.2
TLS+= NOJ*0.19+NS*0.3-NSIM*0.48+MSR*6-1.1
TLS-= NOJ*0.15+NS*0.29-NSIM*0.48+MSR*6-2.1

Large problem

TLS= NOJ*0.47+NS*0.6-NSIM*0.86+MSR*6-1.6
IWI= NOJ*0.59+NS*0.75-NSIM*3.1+MSR*3.7
EIL= NOJ*0.4+NS*0.68+NSIM*0.6-MSR*1.1
TLS+= NOJ*0.15+NS*0.33-NSIM*0.32+MSR*6-0.1
TLS-= NOJ*0.11+NS*0.12-NSIM*0.34+MSR*6-1.9

The search algorithm works on the processing sequence for the first stage. In the following stages, the jobs will be processed according to the FCFS (first-come-first-served) rule. Because of the existence of parallel machines and machine skipping, the processing sequence may vary from stage to stage (non-permutation scheduling). Thus, the beginning sequence may not be the same as the ending sequence. The performance of each beginning sequence is evaluated by the objective function value of the ending sequence resulted from it based on the FCFS rule. To demonstrate the search algorithm, a small example problem is used. In the interest of clarity and space, the example problem is generated using the 3-stage structure (flowshop A). This research is focused on not only solving the problem configuration that exists as of today but also those configurations that can emerge in the future. To represent a possible machine configuration in the future, an unrelated parallel machine is introduced in stage 2. The machine skipping ratio of the example problem is 0.1, which means that each job has a probability of 0.1 to skip processing on a machine. If a job is skipping a machine, the job will have 0 as its run time and setup time on that machine. Table 1 shows the main data of the example problem (Rel: release time of jobs). Table 2 shows the setup time matrix on machine 1 of the example problem (Ref: the reference job). The setup time of job 7 when it is assigned in the 1st position in the schedule, for instance, is “37”, which is located in the 1st row (for jobs processed following the reference job) under column 7. The setup time of job 7 when it is assigned after job 2 in the schedule is “20”, which is located in the 2nd row (for jobs processed following job 2) under column 7.

The tabu search is initialized by adopting the IS as the best solution. The current job-position combination in the LTM matrix is updated to 1. After that, the short term

search initialization is performed by calculating search parameters by the empirically determined formulae (TLS is set to 3, IWI is set to 7, EIL is set to 7), and admitting the current solution to CL and IL. For this problem, there are 45 possible exchange moves and 90 possible insert moves that are going be explored. Among all those moves, the exchange move between position 6 (J7) and position 10 (J8) gives the best objective function value as 1648. Because TL is empty now, this move is not tabu. This exchange move is applied, and then admitted into TL. The best solution is updated and AL is set to 1648. The elements representing the job-position combinations in this new solution in LTM matrix is increased by 1. This sequence is admitted to CL because it has a better objective function value than its parent. Because number of iterations without improvement and entries into the IL are still 0, the stopping criteria is not met and the search will continue by perturbing the job sequence again. In this iteration, the exchange move between position 2 (J3) and position 5(J6) is selected with the objective function value of 1304. After 16 more iterations, the total entries into the IL reaches 7, and the short-term search is stopped. Table 3 lists the iterations during this short-term search.

Table 1. Example Problem

				Stage			
				1	2	2	3
				Machine Availability			
				19	26	48	67
				Run Time			
Job	Weight	Rel	Due Date	M1	M2	M3	M4
1	1	12	206	16	39	36	37
2	3	38	143	2	0	0	13
3	2	3	130	17	16	16	38
4	2	4	224	35	18	23	11
5	1	15	85	0	7	16	20
6	3	2	162	13	40	26	15
7	1	18	178	28	8	19	7
8	3	5	255	19	8	5	19
9	3	20	159	4	30	29	0
10	2	9	186	38	0	0	22

Table 2. Setup Time Matrix on Machine 1

J'\J	1	2	3	4	5	6	7	8	9	10
Ref	26	24	16	28	0	6	37	6	18	36
1	6	35	7	13	0	8	37	34	17	26
2	30	28	8	37	0	17	20	35	14	12
3	20	26	11	6	0	22	38	5	39	5
4	32	17	28	36	0	9	29	22	32	13
5	37	26	15	40	0	28	10	26	29	14
6	40	38	27	37	0	40	37	1	4	34
7	25	33	2	36	0	33	32	19	35	20
8	13	3	20	9	0	36	24	35	20	20
9	38	9	3	21	0	32	19	35	26	33
10	35	39	35	25	0	19	39	21	6	19

The LTM matrix is now used to identify a new sequence and trigger the search again using that sequence. LTM-max identifies the most frequent job-position combination. This combination will be fixed in the next iteration of the search. Table 4 is the LTM matrix.

According to this matrix, Job 3 in position 5 will be fixed during the next iteration of the search. The search will restart twice with the new sequence identified by the LTM matrix and finally return the best solution with the lowest total weighted tardiness. The best solution has been found during the first short-term search, with the best job sequence of [J6, J2, J5, J9, J3, J4, J10, J8, J7, J1] and the total weighted tardiness of 1108.

Table 3. Search Iterations

Solution #	Sequence	Obj Fuc	CL Entry	IL Entry
1	5 3 2 9 6 7 10 1 4 8	2204	Y	Y
2	5 3 2 9 6 8 10 1 4 7	1648	Y	N
3	5 8 2 9 6 3 10 1 4 7	1225	Y	Y
4	5 8 2 9 6 3 1 10 4 7	1268	N	N
5	5 8 2 9 3 6 1 10 4 7	1328	N	N
6	5 6 2 9 3 8 1 10 4 7	1304	Y	N
7	5 6 9 2 3 8 1 10 4 7	1258	Y	Y
8	6 5 9 2 3 8 4 10 1 7	1357	N	N
9	6 5 9 8 2 3 4 10 1 7	1201	Y	Y
10	6 5 9 8 3 2 4 10 1 7	1303	N	N
11	6 5 9 2 3 8 4 10 7 1	1146	Y	Y
12	6 5 2 9 3 8 4 10 1 7	1157	N	N
13	6 5 2 9 3 8 4 1 10 7	1290	N	N
14	6 5 2 9 3 8 4 10 7 1	1113	Y	Y
15	6 5 2 9 3 4 10 8 7 1	1119	N	N
16	6 2 5 9 3 4 10 8 7 1	1108	Y	Y
17	6 2 9 5 3 4 10 8 7 1	1138	N	N
18	6 2 9 5 3 4 10 8 1 7	1117	Y	Y
19	6 2 5 9 3 4 10 8 1 7	1144	N	N

Table 4. LTM Matrix

	Position									
Job	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	4	4	6	5
2	0	4	10	3	1	1	0	0	0	0
3	0	2	0	0	14	3	0	0	0	0
4	0	0	0	0	0	5	7	0	7	0
5	7	8	2	2	0	0	0	0	0	0
6	12	2	0	0	4	1	0	0	0	0
7	0	0	0	0	0	1	0	0	5	13
8	0	3	0	2	0	8	0	5	0	1
9	0	0	7	12	0	0	0	0	0	0
10	0	0	0	0	0	0	8	10	1	0

6. Experimental design

Given this research is inspired by a real industry problem, who's most desired goal is a methodology to schedule jobs with the passage of time, a mathematical programming based approach is not a suitable solution. Besides, the capability of mathematical programming method is limited to solving small problems, even then there is no guarantee that optimal solutions to small problems can be identified within a reasonable computation time as the problem is NP-hard in the strong sense. An efficient and effective procedure (in this case the search algorithm) that is implementable in practice is of significant interest to the company, which is where we devoted our time on to develop. To further analyze the performance of the ISs and algorithms, not only based on different flowshops, as they currently exist, but also going from current to maximum configuration that could conceivably span several years into the future, an extensive statistical experiment was

designed and implemented. Before discussing about the performance of ISs and algorithms, the way of generating different problems need to be demonstrated. The problem size (3 levels) is decided by the number of jobs. As described in Section 1, the base structures are inspired by the current machine-stage layouts. When NSIM is set to 0, a machine-stage layout is generated as the current layout (the current flowshop A, B or C). When NSIM is set to a value bigger than 0, say 2 for instance, 2 stages are randomly selected and additional machines (1 for each stage) are introduced to those stages. Therefore, a combination of structure and NSIM (4 levels for flowshop A, 7 levels for flowshop B, 6 levels for flowshop C) decides a particular machine-stage layout. MSR (3 levels) is the last factor that determines the chances of a job skipping process in a stage. Two replicates were generated within each CP combination. Beside the six ISs, we developed six algorithms to solve the problem: Short term memory with fixed TLS (STM-F), short term memory with variable TLS (STM-V), LTM-max with fixed TLS (L-Max F), LTM-min with fixed TLS (L-Min F), LTM-max with variable TLS (L-Max V), and LTM-min with variable TLS (L-Min V). Thus, $3 \times (4 + 7 + 6) \times 3 \times 2 \times 6 \times 6 = 11016$ results were collected. Table 5 shows the percentage of times an IS/algorithm leads to identifying the best solution (compared to the others) based on different problem sizes and number of stages.

Table 5. ISs/Algorithms Performances on Problem Size

Problem Size	Number of Stages	IS1 (%)	IS2 (%)	IS3 (%)	IS4 (%)	IS5 (%)	IS6 (%)	STM-F (%)	STM-V (%)	L-Max F (%)	L-Min F (%)	L-Max V (%)	L-Min V (%)
Small	3	39	40	93	75	28	12	82	83	90	93	99	84
	5	29	36	94	59	85	22	78	73	79	83	99	85
	6	31	23	84	34	77	43	73	74	74	75	75	95
	Average	33	33	90	56	63	26	78	77	81	84	91	88
Medium	3	59	37	50	87	79	27	36	49	96	62	78	92
	5	73	32	68	82	95	48	57	64	94	62	95	96
	6	89	25	45	79	99	67	66	79	89	67	90	83
	Average	74	31	54	83	91	47	53	64	93	64	88	90
Large	3	60	25	51	92	81	45	25	28	89	63	80	86
	5	67	28	78	95	88	78	32	42	68	42	70	88
	6	75	25	39	99	80	93	51	58	63	53	68	78
	Average	67	26	56	95	83	72	36	43	73	53	73	84

Clearly, different ISs perform differently on problems in different sizes and different number of stages. For problems in all different sizes, IS1, IS2, and IS4 have decreasing ability to give the best result moving from a 3-stage problem (flowshop A) to 6-stage problem (flowshop B). IS5 gives a good result in 5-stage problems (flowshop C). IS6 has an increasing ability to give better results for problems with more stages. Also, different algorithms have different performances on different problems. L-Min is very good at solving 6-stage problems. L-Max V provides good results for 5-stage problems.

Again, the most important characteristic of this research is that we are not only providing solutions for the current layout of machines, but also taking future developments of the company into consideration in being able to provide solutions when the number of stages with increased machines (NSIM) is considered as a factor. Thus, some results are collected based on the change of NSIM. Table 6 gives the percentage of times an IS/algorithm identifies the best result. Both ISs and algorithms perform differently as NSIM increases.

Table 6. Performance of ISs/Algorithms on NSIM

Number of Stages	NSIM	IS1 (%)	IS2 (%)	IS3 (%)	IS4 (%)	IS5 (%)	IS6 (%)	STM-F (%)	STM-V (%)	L-Max F (%)	L-Min F (%)	L-Max V (%)	L-Min V (%)
3	0	65	53	82	96	45	15	63	62	93	71	90	81
	1	57	51	73	91	53	22	50	53	89	66	88	76
	2	49	50	54	79	68	29	47	54	96	70	82	94
	3	40	44	49	75	85	45	33	53	90	68	71	96
5	0	68	56	99	92	81	62	70	65	88	71	96	86
	1	66	55	95	85	85	50	67	70	88	70	98	84
	2	58	40	86	77	88	53	55	59	82	61	93	87
	3	50	39	75	76	94	52	55	62	71	66	90	90
	4	52	25	72	75	98	42	50	47	67	55	92	98
	5	45	23	53	69	99	35	38	55	54	54	99	94
6	0	70	65	95	90	71	85	69	80	90	79	85	75
	1	74	62	81	81	76	76	77	83	91	78	83	78
	2	72	53	74	80	83	74	73	77	87	74	78	83
	3	68	45	60	70	86	68	72	75	82	73	73	88
	4	67	42	55	65	90	61	56	66	72	64	69	86
	5	58	40	31	60	91	58	50	67	65	52	62	91
	6	46	29	20	51	98	55	45	70	65	54	63	100

The results reported above are based upon numerical values with no statistical validity. To test the significance of the difference between ISs and algorithms, a statistical experiment is designed.

To identify the difference between ISs and algorithms in solving different problems, the PCPs (problem characteristic parameters) are designed to be main factors in the split plot design, while algorithm and IS are put in the subplot. Moreover, NSIM is nested with NS because for different stage-machine layouts structures, NSIM have different ranges. Thus, the statistical experiment falls into the category of split-plot and nested design as described in Montgomery (2009). The null hypotheses of interest in this research are:

- Quality of the ISs and search algorithms is the same
- Performances of the ISs and search algorithms in different problem sizes are the same
- Performances of the ISs and search algorithms in different structures are the same
- Performances of the ISs and search algorithms with different MSRs are the same
- Performances of the ISs and search algorithms with different NSIMs are the same
- Performances of the algorithms with different ISs are the same

The statistical model can be formulated as given below:

$$\begin{aligned} y_{ijklmnpq} = {} & \mu + \tau_i + \beta_j + \gamma_k + \theta_n + \delta_l + \omega_{m(l)} + (\gamma\theta)_{kn} + (\gamma\delta)_{kl} + (\gamma\omega)_{km(l)} + (\theta\delta)_{nl} \\ & + (\theta\omega)_{nm(l)} + (\gamma\theta\delta)_{knl} + (\gamma\theta\omega)_{knm(l)} + \varepsilon_1 + \sigma_q + \varphi_p + (\gamma\sigma)_{kq} + (\gamma\varphi)_{kp} \\ & + (\theta\sigma)_{nq} + (\theta\varphi)_{np} + (\delta\sigma)_{lq} + (\delta\varphi)_{lp} + (\omega\varphi)_{m(l)p} + (\omega\sigma)_{m(l)q} + (\sigma\varphi)_{qp} \\ & + (\gamma\theta\sigma)_{knq} + (\gamma\theta\varphi)_{knp} + (\gamma\delta\sigma)_{klq} + (\gamma\delta\varphi)_{klp} + (\gamma\omega\sigma)_{km(l)q} + (\gamma\omega\varphi)_{km(l)p} \\ & + (\gamma\varphi\sigma)_{kpq} + (\theta\delta\sigma)_{nlq} + (\theta\delta\varphi)_{nlp} + (\theta\omega\sigma)_{nm(l)q} + (\theta\omega\varphi)_{nm(l)p} + (\theta\sigma\varphi)_{nqp} \\ & + (\delta\sigma\varphi)_{lqp} + (\omega\sigma\varphi)_{m(l)qp} + (\gamma\theta\delta\sigma)_{knlq} + (\gamma\theta\delta\varphi)_{knlp} + (\gamma\theta\omega\sigma)_{kmn(l)q} \\ & + (\gamma\theta\omega\varphi)_{knm(l)p}(\gamma\theta\sigma\varphi)_{knqp} + (\gamma\delta\sigma\varphi)_{klqp} + (\gamma\omega\sigma\varphi)_{km(l)qp} + (\theta\delta\sigma\varphi)_{nlqp} \\ & + (\theta\omega\sigma\varphi)_{nm(l)qp} + (\gamma\theta\delta\sigma\varphi)_{knlqp} + (\gamma\theta\omega\sigma\varphi)_{knm(l)qp} + \varepsilon_2 \end{aligned}$$

$$\begin{cases} j = 1, 2 \\ k = 1, 2, 3 \\ n = 1, 2, 3 \\ m = 0, 1, 2, 3 \ (l = 1), \quad m = 0, 1, 2, 3, 4, 5 \ (l = 2), \quad m = 0, 1, 2, 3, 4, 5, 6 \ (l = 3) \\ q = 1, 2, 3, 4, 5, 6 \\ p = 1, 2, 3, 4, 5, 6 \end{cases}$$

Where β_j is the replicate, γ_k is the problem size, θ_n is the MSR, δ_l is the structure, $\omega_{m(l)}$ is the NSIM (nested with structure), σ_q is the IS, φ_p is the algorithm, ε_1 is the whole plot error, and ε_2 is the subplot error.

The ANOVA is performed by Statgraphics Centurion XV (2010). The distribution of the actual values of total weighted tardiness turned out to be non-normal. In this situation, a nonparametric ranking method is recommended to be used (Montgomery 2009) to perform Kruskal-Wallis Test. In this experiment, the results form ranked and original values are not the same (2^{nd} and 3^{rd} columns in Table 7). Both of the values showed that the difference between the quality of the search algorithms is significant, but the difference between the quality of the ISs is reported to be insignificant by the ranked ANOVA. In this case, data transformation (other than ranked transformation) is recommended to pursue a more accurate prediction of the significance of the interactions. The usual transformations are inversion and LOG_{10}. The LOG_{10} transformation turned out to be better than inversion but it was still not providing a satisfactory result. After several attempts, ($LOG_{10}^{1.2}$) is used to transform the original values of total weighted tardiness to ensure normality. The power of the test is above 95%, confirming that two replicates are adequate. The result is shown in the 1^{st} column of Table 7. The transformed data confirmed that the interaction between search algorithms/ISs, the structure, and size of problem are significant. Also, the interaction between algorithms/ISs and NSIM (nested in structure) is significant. Those conclusions also support the numerical findings reported above in Tables 5 and 6.

To uncover the difference between the quality of search algorithms and ISs, Tukey's test is used. There are several interactions that reported to be significant. Based on the significant interactions, the test is performed once by fixing the problem size at small, medium, and large, then varying structures, and once by fixing the structure at 3 stages, 5 stages, and 6 stages, and then varying NSIM from 0 to the maximum accordingly, which gives possible machine configurations in the future. The results show that IS3 is the best for small size problems; IS4 is the best for medium size problems with 3 stages, while IS5 is the best for problems with both 5 and 6 stages; IS4 is the best for large-size problems, while IS6 is comparable with IS4 for 6-stage large size problems. In 3-stage problems, IS4 is the best when NSIM is 0, 1, or 2, while IS5 is the best when NSIM is 3. In 5-stage problems,

Table 7. ANOVA for the split plot design

		Transformed TWT		Ranked TWT		Original TWT	
Source	DOF	F-Ratio	P-Value	F-Ratio	P-Value	F-Ratio	P-Value
Structure	2	4.92E-02	0.05	1.74E-01	0.16	2.74E-01	0.24
Size	2	3.91E-02	0.04	1.22E-02	0.01	1.22E-03	0
MSR	2	2.17E-01	0.19	3.57E-01	0.3	3.80E-02	0.04
NSIM(Structure)	14	2.53E+00	1	8.74E-01	0.41	1.74E-01	0
Replicate	1	3.57E-01	0.45	4.19E-01	0.48	8.02E-01	0.63
Structure*size	4	1.13E-01	0.02	1.90E-01	0.06	1.13E-01	0.02
Structure*MSR	4	1.61E-01	0.04	1.61E-01	0.04	1.61E-01	0.04
size*MSR	4	9.01E-01	0.54	9.70E-01	0.58	9.70E-01	0.58
size*NSIM(Structure)	28	2.92E-01	0	3.58E-01	0	1.49E-01	0
MSR*NSIM(Structure)	28	2.22E+00	1	1.98E-01	0	1.52E+00	0.95
Structure*size*MSR	8	2.39E+00	0.98	3.22E-01	0.04	1.35E+00	0.78
size*MSR*NSIM(Structure)	56	2.26E+00	1	8.20E-01	0.18	9.53E-01	0.43
Main plot error	404						
algorithm	5	7.37E-02	0	8.53E-02	0.01	3.12E-02	0
IS	5	1.68E+00	0.87	4.76E+00	1	2.07E-01	0.04
Structure*algorithm	10	8.91E-04	0	8.27E-01	0.4	1.28E+00	0.76
size*algorithm	10	4.97E-03	0	1.39E-02	0	5.75E-03	0
MSR*algorithm	10	5.23E-01	0.12	2.64E-03	0	4.37E-04	0
NSIM(Structure)*algorithm	70	6.26E-01	0.01	1.27E+00	0.93	1.38E+00	0.98
Structure*size*algorithm	20	1.58E-03	0	5.91E-03	0	2.00E-03	0
Structure*MSR*algorithm	20	2.98E+00	1	4.27E+00	1	4.10E+00	1
size*MSR*algorithm	20	1.58E+00	0.95	1.39E+00	0.89	4.21E-01	0.01
Structure*IS	10	7.83E-02	0	5.99E-02	0	7.30E-02	0
size*IS	10	9.41E-02	0	6.58E-01	0.24	1.36E+00	0.81
MSR*IS	10	8.90E-01	0.46	3.96E-02	0	1.14E-01	0
NSIM(Structure)*IS	70	1.16E-01	0	8.32E-01	0.16	8.61E-01	0.21
Structure*size*IS	20	4.39E-01	0.01	7.96E-02	0	6.77E-02	0
Structure*MSR*IS	20	5.76E-01	0.07	8.97E-01	0.41	5.09E-01	0.04
size*MSR*IS	20	5.87E-01	0.08	7.83E-01	0.26	8.55E-01	0.35
algorithm*IS	25	7.93E-03	0	1.64E-02	0	1.10E-02	0
Structure*algorithm*IS	50	1.32E+00	0.94	1.38E+00	0.96	1.17E+00	0.81
size*algorithm*IS	50	1.04E+00	0.6	1.28E+00	0.91	1.99E+00	1
MSR*algorithm*IS	50	8.70E-01	0.27	8.98E-01	0.32	8.49E-01	0.23
Sub plot Error	10343						
Total (corrected)	11015						

IS3 is the best when NSIM is 0 and 1, there is no difference between IS3 and IS5 when NSIM is 3, but when NSIM is 4 or 5, IS5 turned out to be the best. In 6-stage problems, IS3

and IS4 outperform the others when NSIM is 0 or 1, and IS3, IS4, and IS5 outperform others when NSIM is 2. IS5 is the best when NSIM is 3, 4, 5, or 6.

In small problems, L-Max V is the best in 3-stage and 5-stage problems, while L-Min V is the best in 6-stage problems. In medium problems, L-Max F and L-Min V perform the same on 3-stage and 5-stage problems (better than the others) while L-Max V performs best on 6-stage problems. In large problems, L-Min F performs the best on 3-stage problems, L-Min V performs the best on 5-stage problems, and L-Max V performs the best on 6-stage problems. Based on number of stages, in 3-stage problems, there is no difference between L-Max F and L-Max V in identifying the best solutions when NSIM is 0 or 1. L-Max F performs the best when NSIM is 2 or 3. In 5-stage problems, L-Max V and L-Min V perform the same (while outperforming the others) in identifying the best solutions with all possible NSIMs. In 6-stage problems, L-Max F performs the best when NSIM is varied from 0 to 2. When NSIM is varied from 3 to 6, L-Min V performs the best. Based on ISs, L-Max F is the best algorithm for IS3, while L-Min V is the best algorithm for IS5. For the other ISs, long-term memories have a better performance than short-term memories.

7. Conclusions and future research

In this paper, a research problem closely guided by an industry problem has been investigated. A search algorithm based on Tabu search was developed to solve this hybrid flowshop. The goal is to minimize total weighted tardiness of all jobs. The most challenging part of this research is to develop an algorithm that not only works for the current machine layout but also for all possible machine layouts in the future. The performance of the ISs and algorithms are tested by using randomly generated example problems. A statistical analysis was performed based on nested split-plot design. The results show a statistically significant difference between ISs/algorithms and problem sizes, structures, NSIMs, which means that given a problem size, structure, and NSIM, there will be at least one IS and one algorithm that are proven to be better than the others in solving this problem The numerical evaluation of the percentage of times when an IS/algorithm leads to the best result reconciled with the findings of the statistical tests. Based on the results from Tukey's test, best ISs and algorithms are identified for different kind of

problems. It is interesting to see that both the performance of algorithms and ISs show some trend when NSIM increases, which means that the best IS/algorithm for the company now may not necessarily be the best one in the future. Future research can focus on the development of a mathematical model for this problem in order to evaluate the quality of the solutions produced by the ISs/algorithms developed in this research by optimally solving small problem instances to the extent possible.

References:

Alisantoso, D., Khoo, L., & Jiang, P. (2003). An immune algorithm approach to the scheduling of a flexible PCB flow shop. *International Journal of Advanced Manufacturing Technology, 22*(11), 819–827.

Allahverdi, A., Gupta, J.N.D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega, The International Journal of Management Science, 27*, 219-239.

Bilge, U., Kirac, F., Kurtulan, M., & Pekgun, P. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research, 31*(3), 397-414.

Biskup, D., Herrmann, J., & Gupta, J.N.D. (2008). Scheduling identical parallel machines to minimize tardiness. *International Journal of Production Economics, 115*(1), 134-142.

Chen, B., Potts, C.N., & Woeginger, G.J. (1998). *Handbook of Combinatorial Optimization, A review of machine scheduling: complexity, algorithms and applications.* Dordrecht, Netherlands: Kluwer.

Dogramaci, A., & Surkis, J. (1979). Evaluation of a heuristic for scheduling independent jobs on parallel identical processors. *Management Science, 25*(12), 1208–1216.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Journal of Computers and Operations Research, 13*,533–549.

Gupta, J.N.D., Krüger, K., Lauff, V., Werner, F., & Sotskov, Y.N. (2002). Heuristics for hybrid flow shops with controllable processing times and assignable due dates. *Computer Operational Research, 29*(10), 1417–1439.

Lee, Y., Bhaskran, K., & Pinedo, M. (1997). A heuristic to minimize total weighted tardiness with sequence dependent setups. *IIE Transactions, 29*(1), 45–52.

Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.J., & Shmoys, D.B. (1993). *Handbooks in Operations Research and Management Science, Logistic of Production and Inventory: Vol. 4. Sequencing and scheduling: algorithm and complexity*. Amsterdam, Netherlands: North-Holland.

Logendran, R., McDonell, B., & Smucker, B. (2007). Scheduling unrelated parallel machines with sequence dependent setups. *Computer & Operational Research, 34*(11), 3420-3438.

Logendran, R., & Sonthinen, A. (1997). A Tabu search-based approach for scheduling job-shop type flexible manufacturing systems. *Journal of the Operational Research Society, 48*, 264-277.

Logendran, R., & Subur, F. (2004). Unrelated parallel machine scheduling with job splitting. *IIE Transactions, 36*, 359-372.

Montgomery, D.C. (2008). *Design and Analysis of Experiments*. NY: John Wiley & sons.

Oakdale Engineering (2008). *DATAFIT 9.059*, PA, USA

Pandya, V., & Logendran, R. (2010). Weighted Tardiness Minimization in Flexible Flowshops. *Proceedings of the 2010 Industrial Engineering Research Conference*

Rubin P.A., & Ragatz G.L. (1995). Scheduling in a sequence dependent setup environment with genetic search. *Computers and Operations Research, 22*(1), 85-99.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*, 781-800.

Sen, T., Sulek, J.M., & Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *International Journal of Production Economics, 83*(1), 1–12.

Shim, S., & Kim, Y. (2007). Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Production Research, 177*(1), 135-146.

Stat Point Technologies Inc. (2010). *STATGRAPHICS Centurion XVI, Version 16.1.02*, VA, USA.

Tan, K. C., & Narasimhan, R. (1997). Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach. *Omega, 25*(6), 619–634.

Wilkerson, L.J., & Irwin, J.D. (1971). An improved algorithm for scheduling independent tasks. *AIIE Transactions, 3*(1), 239–245.

Zhang, G.H., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job shop scheduling problem. *Expert Systems with Applications, 38*(4), 3563–3573.